**Handwritten Document Recognition System**

**By Lakshmi Vyshnavi Ummadisetti**

**URL:**[**https://drive.google.com/drive/folders/1We6u0zaGxW7ns2WAxt6eGgjLWKdmnhIb?usp=sharing**](https://drive.google.com/drive/folders/1We6u0zaGxW7ns2WAxt6eGgjLWKdmnhIb?usp=sharing)

**Abstract**

This project implements a Handwritten Document Recognition System designed to automatically detect, extract, and interpret handwritten text from images. Utilizing a combination of image preprocessing, text region detection, and Optical Character Recognition (OCR) technologies, the system achieves a measurable performance evaluated through metrics like Character Error Rate (CER) and Word Error Rate (WER). The findings demonstrate significant areas for optimization in recognizing complex handwritten text.

**Introduction**

In today's digitized environments, the ability to process and digitize handwritten documents is a critical AI-driven service. From indexing lecture notes to cataloguing archives, handwritten document recognition enables accessibility and automation in various domains.

**Objectives**

1. Automatically detect and extract handwritten text regions from the given input image.
2. Recognize and convert extracted text into machine-readable format.
3. Evaluate the system's performance using appropriate metrics.

**Methodology:**

1. **Preprocessing**

**Techniques Used:**

* **Grayscale Conversion:** Simplifies image processing by reducing the image to a single channel and enhancing contrast.
* **Noise Reduction:** Gaussian blur was applied to eliminate high-frequency noise that could interfere with text detection.
* **Thresholding:** Adaptive thresholding was used to enhance text visibility by converting grayscale images into binary format. This step improves the accuracy of text region detection.

**Why These Techniques?**

* Grayscale conversion reduces computational complexity.
* Noise reduction ensures clean input for subsequent steps.
* Thresholding makes text boundaries clearer for detection algorithms.

**Code Reference:** preprocess.py

1. **Text Region Detection**

**Method:** EAST (Efficient and Accurate Scene Text Detector)

* **Pipeline:** The EAST model detects text regions in images by predicting both the text score map and geometry.
* **Bounding Boxes:** Non-Maximum Suppression (NMS) was applied to refine overlapping predictions and identify clear text regions.

**Challenges:**

* Text regions with uneven illumination or overlapping characters required fine-tuning of the detection parameters.

**Why EAST?**

* Provides a balance between accuracy and computational efficiency.
* Handles multi-scale text detection effectively.

**Code Reference:** detect\_text.py

1. **OCR for Recognition**

**Tool:** EasyOCR

* EasyOCR was used to recognize text from detected regions. It employs a deep learning-based approach combining convolutional and recurrent layers.
* The tool was configured to handle variations in handwriting styles.

**Why EasyOCR?**

* Pre-trained model with support for multiple languages and handwriting.
* Easy integration into Python-based workflows.

**Comparison with Alternatives:**

* Tesseract OCR struggled with handwritten text due to its rule-based engine.
* EasyOCR demonstrated higher accuracy for this task.

**Code Reference:** recognize\_text.py

1. **Evaluation**

**Metrics Used:**

1. **Character Error Rate (CER):** Quantifies character-level discrepancies to evaluate fine-grained accuracy.
2. **Word Error Rate (WER):** Measures word-level differences, highlighting issues in contextual recognition.

**Why These Metrics?**

* CER focuses on individual character recognition accuracy, essential for handwritten text.
* WER evaluates the system's ability to maintain word integrity, critical for preserving meaning.

**Code Reference:** evaluate.py

**Architectural Design**

The architectural design of the Handwritten Document Recognition System integrates several components into a streamlined pipeline. The process is divided into four main stages:

1. **Image Preprocessing**
2. **Text Region Detection**
3. **Text Recognition**
4. **Evaluation**

**Pipeline Stages**

**1. Image Preprocessing**

* **Purpose:** Prepare raw input images for accurate text detection and recognition.
* **Steps:**
  1. Convert images to grayscale.
  2. Apply noise reduction using Gaussian blur.
  3. Enhance contrast through adaptive thresholding.
* **Output:** Cleaned and binarized images ready for region detection.

**2. Text Region Detection**

* **Purpose:** Identify areas in the image likely containing handwritten text.
* **Steps:**
  1. Use the EAST model to predict text score maps and geometric data.
  2. Apply Non-Maximum Suppression (NMS) to refine bounding box predictions.
* **Output:** Bounding boxes highlighting text regions.

**3. Text Recognition**

* **Purpose:** Extract and convert handwritten text into machine-readable format.
* **Steps:**
  1. Crop detected text regions.
  2. Pass cropped regions through EasyOCR for recognition.
* **Output:** Recognized text strings.

**4. Evaluation**

* **Purpose:** Measure the system's accuracy and performance.
* **Steps:**
  1. Compare recognized text to ground truth.
  2. Compute Character Error Rate (CER) and Word Error Rate (WER).
* **Output:** Performance metrics quantifying recognition accuracy.

**Model Selection**

**EAST Model for Text Detection**

* **Why Chosen:**
  + Efficient multi-scale text detection.
  + Lightweight and integrates well with real-time systems.
* **Alternatives Considered:**
  + YOLO-based detectors (less specialized for text).

**EasyOCR for Text Recognition**

* **Why Chosen:**
  + Pre-trained on diverse handwriting datasets.
  + Combines convolutional and recurrent layers for robust sequence modeling.
* **Alternatives Considered:**
  + Tesseract OCR (lower accuracy on handwriting).

**Evaluation of the Result:**

**Metrics:**

* **Character Error Rate (CER): 2.47%**
  + Indicates that only 2.47% of the characters in the recognized text were either incorrect, missing, or extra when compared to the ground truth.
  + This low CER highlights strong character-level accuracy, suggesting that preprocessing and OCR effectively handled the input handwriting.
* **Word Error Rate (WER): 11.94%**
  + Reflects that approximately 11.94% of the words in the recognized text were incorrect or mismatched.
  + A relatively low WER suggests good word-level recognition but also shows that there is some room for improvement, possibly in handling word boundaries or contextual relationships.

**Rationale for Chosen Models and Techniques**

The selection of models and techniques was guided by a balance of accuracy, computational efficiency, and ease of integration into the pipeline.

* Grayscale conversion reduces computational complexity and highlights text regions by eliminating unnecessary color information. Gaussian blur minimizes high-frequency noise, improving the clarity of text edges. Adaptive thresholding ensures consistent text visibility, even in images with uneven illumination.

**EAST Model for Text Detection**

* EAST (Efficient and Accurate Scene Text Detector) provides state-of-the-art performance for text detection, particularly in scenarios involving multi-scale and irregularly shaped text.
* It employs a fully convolutional network, making it lightweight and efficient, suitable for both real-time applications and large-scale batch processing.
* Its ability to produce both text score maps and geometry for bounding boxes enables precise localization of text regions, which is critical for accurate OCR.

**EasyOCR for Text Recognition**

* EasyOCR combines convolutional and recurrent neural networks, leveraging deep learning to handle the sequential nature of text.
* Pre-trained on diverse datasets, it offers robust performance for handwritten text recognition, outperforming traditional OCR tools like Tesseract in accuracy.
* Its support for multiple languages and handwriting styles made it an excellent choice for generalizability.